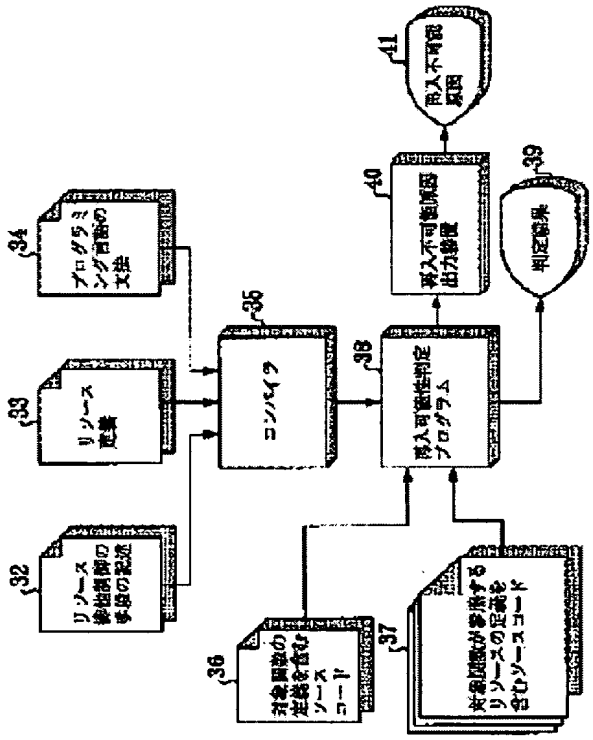
METHOD FOR DECIDING REINPUT POSSIBILITY OF FUNCTION

Patent number:	JP2001134431
Publication date:	2001-05-18
Inventor:	OMORI MASANORI
Applicant:	MITSUBISHI ELECTRIC CORP
Classification:	
- international:	G06F9/06; G06F11/28
- european:	
Application number:	JP19990318181 19991109
Priority number(s):	JP19990318181 19991109

Report a data error here

Abstract of **JP2001134431**

PROBLEM TO BE SOLVED: To improve a processing speed and decision reliability by automatically deciding the reinput possibility of a function. SOLUTION: This device is provided with a compiler 35 for generating a reinput possibility decision program 38 from a grammar 34 of programming language, resource definition 33, and description 32 of a resource exclusion controlling means, and a reinput possibility decision program 38, and a reinput impossibility cause outputting device 40.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2001－134431
（P2001－134431A）
(43)公開日　平成13年５月18日(2001.5.18)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
G06F	9/06	540	G06F	9/06	540T　5B033
	9/40	310		9/40	310A　5B042
	9/45			9/46	340F　5B076
	9/46	340		11/28	E　5B081
	11/28			9/44	322C　5B098

審査請求　未請求　請求項の数４　ＯＬ　（全９頁）

(21)出願番号　特願平11－318181

(22)出願日　平成11年11月９日(1999.11.9)

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号

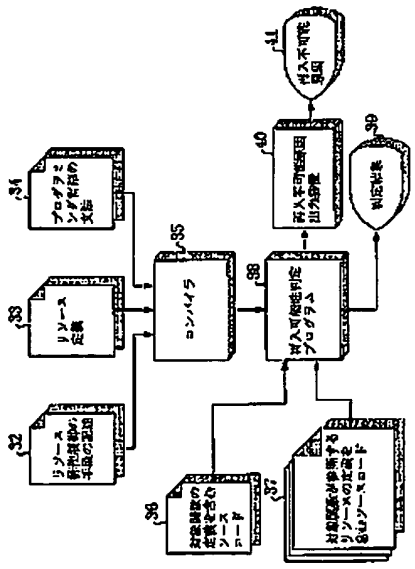
(72)発明者　大森　正則
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

(74)代理人　100102439
弁理士　宮田　金雄　（外２名）

Ｐターム(参考)　5B033 DA03
5B042 HH10
5B076 EC03
5B081 CC11 CC41
5B098 GA02 GA04 GA06 GD15

(54)【発明の名称】　関数の再入可能性判定方法

(57)【要約】
【課題】　関数の再入可能性を自動的に判定し，処理速度と，判定の信頼性を向上させる。
【解決手段】　プログラミング言語の文法３４と，リソース定義３５，リソース排他制御の手段の記述３４から，再入可能性判定プログラム３８を生成するコンパイラ３５と，再入可能性判定プログラム３８と再入不可能原因出力装置４０を設けた。



1

【特許請求の範囲】

【請求項１】　時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムと、関数型プログラミング言語と、前記オペレーティングシステムが提供するリソースの排他制御機能、または、前記オペレーティングシステムに対して提供されたリソースの排他制御機能を持つライブラリを用いることを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を、前記プログラミング言語の文法と、リソース排他制御の手段の記述と、リソース定義から自動的に判定することを特徴とする関数の再入可能性判定方法。

【請求項２】　前記プログラミング言語が規定する関数の再入可能性を判定することに加えて、判定結果が不可であった場合に判定結果が不可である原因を出力することを特徴とする請求項１記載の関数の再入可能性判定方法。

【請求項３】　前記時分割または、優先度プリエンプティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムがリソースの排他制御機能を提供しない、または、前記オペレーティングシステムに対して、リソースの排他制御機能を提供するライブラリが存在しないことを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を判定する請求項１記載の関数の再入可能性判定方法。

【請求項４】　前記時分割または、優先度プリエンプティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムがリソースの排他制御機能を提供しない、または、前記オペレーティングシステムに対して、リソースの排他制御機能を提供するライブラリが存在しないことを条件として制作するプログラムに包含される前記プログラミング言語が規定する関数の再入可能性を判定することに加えて、判定結果が不可であった場合に判定結果が不可である原因を出力することを特徴とする請求項１記載の関数の再入可能性判定方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、プログラムの信頼性を高めるために利用される、関数型プログラミング言語が規定する関数の再入可能性を判定する方法に関する。

【０００２】

【従来の技術】時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能及びそれに類する機能を有するオペレーティングシステムと関数型プログラミング言語を用いることを条件として制作されるプログラムにおいては、複数のタスクまたは、プロセスからある一定の時間間隔の中で同時に呼び出される関数は再入可能である必要がある。その理由は、関数の動作が予測不能となるためである。従来、関数の再入可能性の判定は人間が手動で行っていたため、判定に時間がかかり、また、判定の信頼性が十分ではなかった。

【０００３】

【発明が解決しようとする課題】時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能を有するオペレーティングシステムと関数型プログラミング言語を用いることを条件として制作されるプログラムにおいて、関数の再入可能性は定義できる。ある関数を複数のタスクまたはプロセスから、その関数仕様を逸脱する動作をすることなしに同時に呼び出すことができれば、その関数は再入可能であるという。関数仕様を逸脱する動作は、グローバルリソース（複数のタスクコンテキストで共有されるリソース）へのアクセスが適切に排他制御されていない場合に起こる。関数が再入可能ではない場合、タスクまたはプロセスの動作が不安定になる可能性が高く、不具合が生じた場合に、不具合の修正作業が極めて困難になるという特徴がある。手動による判定では判定を行おうとする個人の技術によって判定の精度が異なる可能性があり、また、完全な判定である保証はない。さらに、手動による判定では判定に時間がかかるという問題がある。この発明の目的は、前記問題点を解決するために自動で関数の再入可能性の判定を行うことにある。

【０００４】

【課題を解決するための手段】この発明においては、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文とリソースの排他制御を行う手段の記述と、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文とリソースの排他制御を行う手段の記述と入力として、関数の再入可能性を自動判定するプログラムを生成するコンパイラを用いて、関数の再入可能性を自動判定するプログラムを生成し、生成されたプログラムの入力を、判定しようとする関数の定義を含むソースコード及び、対象関数からアクセスされるリソースの定義を含むソースコードとし、判定結果を出力するための装置を出力として、生成されたプログラムを動作させることによって、前記課題を解決する。

【０００５】さらに詳細に記述すると、前記関数の再入可能性を自動判定するプログラムを生成するコンパイラによって生成された、関数の再入可能性を自動判定するプログラムは、判定を行う対象である関数が定義されたソースコードが使用しているプログラミング言語の構文にしたがって、入力されたソースコードを構文解析し、再入可能性判定を行う対象である関数内でアクセスされるすべてのリソースに対して、スコープを決定し、再入可能性判定を行う対象である関数内でアクセスされるリ

ソースのスコープの種類に応じて判定の手段を決定し、前記判定の手段を用いて判定を行う。再入可能性判定を行う対象である関数内でアクセスされるすべてのリソースに対して、判定が正であれば、当該関数は再入可能であると判定される。

【０００６】

【発明の実施の形態】実施の形態１．図１はこの発明のハードウェア構成を示すブロック図．図２はこの発明の実施の形態１におけるソフトウェア構成を示すブロック図．図３は関数の再入可能性判定の処理手順を示すフローチャート、図４はリソースの排他制御の手段を表す模式図である。

【０００７】図２に示す．この発明の実施の形態１を示すソフトウェア構成において、再入可能性判定プログラムを出力するコンパイラ１１は、図１に示すディスク装置５に格納されており、入力装置７から起動されることにより、メモリ３にロードされて中央演算処理装置２において動作を開始し、同じく入力装置７から指定された．ディスク装置５にファイルとして格納されているプログラミング言語の文法１０と、リソース定義９と、リソース排他制御の手段の記述８をＩ／Ｏ制御装置４を介してメモリ３上に読み出し．これを入力として．再入可能性判定プログラム１４をメモリ３上に生成し．起動する。

【０００８】さらに、再入可能性判定プログラム１４は．中央演算処理装置２において動作し、判定しようとする関数の定義を含むソースコード１２及び、対象関数が参照するリソースの定義を含むソースコード１３をディスク装置５から入力し．再入可能性判定結果１５を、出力装置６にＩ／Ｏ制御装置４を介して出力する。図３のフローチャートは、この発明の実施の形態１における再入可能性を判定するプログラム１４の動作を示している。実施の形態１における再入可能性を判定するプログラムは、まず．処理手順１６において．対象関数を解析し．判定対象の関数がアクセスするリソースのリストを作成する（処理手順１７）。次に、処理手順１８において．対象関数が記述されたプログラミング言語の言語仕様におけるスコープの種類に従って．アクセスリソースリストをローカルリソースリストと．グローバルリソースリストに分割する。アクセスするリソースがすべてローカルリソースであれば．すなわち．グローバルリソースリストが空集合であれば．対象関数が複数のタスク、または、プロセスから呼び出されても競合が発生しないので．対象関数は再入可能である（判定１９）。

【０００９】グローバルリソースリストが空集合でない場合は、そのリソース一つ一つに対して適切に排他制御なされているか判定する（判定２１）必要がある。対象となるリソースは図４に示す、関数定義２６において、排他制御に入る処理２７．及び、排他制御から出る処理２８に囲まれた排他制御領域２９に存在しなければ、適切に排他制御されていない。図４においては、グローバルリソースａへのアクセス３０は排他制御されているが．グローバルリソースｂへのアクセス３１は排他制御されていない。よって、グローバルリソースａは図４に示す以外にアクセスがなければ、適切に排他制御されているといえる。図３のフローチャートにおいて．処理手順２０～２４までの処理をグローバルリソースリスト中のすべてのリソースに対して適用すると、ＯＫリスト、ＮＧリストが生成される。ここで、ＮＧリストが空集合であれば、すべてのグローバルリソースへのアクセスは適切に排他制御されているということであるから．対象関数は再入可能であると判定される。ＮＧリストがひとつでも要素を含んでいれば．再入可能ではないと判定される（判定２５）。

【００１０】実施の形態２．図５に示す、この発明の実施の形態２を示すソフトウェア構成において、コンパイラ３５はディスク装置５に格納されており、入力装置７から起動されることにより．メモリ３にロードされ、中央演算処理装置２において動作を開始し、同じく入力装置７から指定された、ディスク装置５にファイルとして格納されている．プログラミング言語の文法３４と、リソース定義３３と、リソース排他制御の手段の記述３２をＩ／Ｏ制御装置４を介してメモリ３上に読み出し、これを入力として．再入可能性判定プログラム３８を生成し．起動する。再入可能性判定プログラム３８は中央演算処理装置２において動作し、判定しようとする関数の定義を含むソースコード３６及び、対象関数が参照するリソースの定義を含むソースコード３７を入力として、再入可能性判定結果３９と．判定が不可の場合に．再入不可能原因出力装置４０を通して、対象関数が再入可能でない原因４１をＩ／Ｏ制御装置４を介して、出力装置６に出力する。

【００１１】図６のフローチャートにおける、処理手順４２～５１の処理に関しては実施の形態１の動作を記述した図３のフローチャートと同一である。実施の形態２では．実施の形態１に加えて、ＮＧリストに存在するグローバルリソースへの判定対象関数におけるアクセス箇所を．対象関数が再入可能でない原因４１として．出力する（処理手順４５）。

【００１２】実施の形態３．図７に示す、この発明の実施の形態３を示すソフトウェア構成において、コンパイラ５６はディスク装置５に格納されており、入力装置７から起動されることにより．メモリ３にロードされて、中央演算処理装置２において動作を開始し、同じく入力装置７から指定された、ディスク装置５にファイルとして格納されているプログラミング言語の文法５５と、リソース定義５４を入力として、再入可能性判定プログラム５９をメモリ３上に生成し、これを起動する。再入可能性判定プログラム５９は中央演算処理装置２において動作し、判定しようとする関数の定義を含むソースコー

ド５７及び、対象関数が参照するリソースの定義を含むソースコード５８を入力として、再入可能性判定結果５３を出力装置６にＩ／Ｏ制御装置４を介して出力する。

【００１３】図８のフローチャートは、この発明の実施の形態３における再入可能性を判定するプログラム５９の動作を示している。実施の形態３では、リソースの排他制御の手段が提供されない場合であるから、処理手順６３において、アクセスリソースリストをローカルリソースリストと、グローバルリソースリストに分割した時点で、グローバルリソースリストが空集合でなければ、再入可能ではないと決定することができる（判定６４）。

【００１４】実施の形態４．図９に示す、この発明の実施の形態４を示すソフトウェア構成において、コンパイラ６７はディスク装置５に格納されており、入力装置７から起動されることにより、メモリ３にロードされて、中央演算処理装置２において動作を開始し、同じく入力装置７から指定された、ディスク装置５にファイルとして格納されているプログラミング言語の文法６６と、リソース定義６５を入力として、再入可能性判定プログラム７０をメモリ３上に生成し、起動する。再入可能性判定プログラム７０は中央演算処理装置２において動作し、判定しようとする関数の定義を含むソースコード６８及び、対象関数が参照するリソースの定義を含むソースコード６９を入力として、再入可能性判定結果７１と、判定が不可の場合に、再入不可能原因出力装置７２を通して、対象関数が再入可能でない原因７３を出力装置６にＩ／Ｏ制御装置４を介して出力する。

【００１５】図１０のフローチャートにおける、処理手順７４～７６に関しては実施の形態３の動作を記述した図８の処理手順６１～６３と同一である。実施の形態４では、実施の形態３に加えて、グローバルリソースリストに存在するリソースへの対象関数におけるアクセス箇所を、対象関数が再入可能でない原因として出力する（処理手順７８）。

【００１６】

【発明の効果】第１の発明は、関数の再入可能性判定プログラム１４により、判定処理の自動化を行うことができ、処理速度の向上と、判定の信頼性の向上を得ることができる。また、プログラミング言語の文法１０、リソースの定義９、リソース排他制御の手段の記述８から関数の再入可能性を判定するプログラムを生成するコンパイラ１１を設けることにより、プログラミング言語、リソース、排他制御の手段のうちひとつまたは複数が異なる環境においても容易に判定処理の自動化を提供することができる。

【００１７】また、第２の発明は、第１の発明の効果に加えて、対象関数が再入可能でなかった場合に、再入不可能原因を出力する装置４０を設けることにより、再入可能ではない原因を特定することができる。

【００１８】また、第３の発明は、リソースの排他制御の手段が提供されなかった場合に、関数の再入可能性判定プログラム５９により、判定処理の自動化を行うことができ、処理速度の向上と、判定の信頼性の向上を得ることができる。また、プログラミング言語の文法５５、リソースの定義５４から関数の再入可能性を判定するプログラム５９を生成するコンパイラ５６を設けることにより、プログラミング言語、リソース、排他制御の手段のうちひとつまたは複数が異なる環境においても容易に判定処理の自動化を提供することができる。

【００１９】また、第４の発明は、第３の発明の効果に加えて、対象関数が再入可能でなかった場合に、再入不可能原因を出力する装置７２を設けることにより、再入可能ではない原因を特定することができる。

【図面の簡単な説明】

【図１】　この発明のハードウェアブロック図である。

【図２】　この発明の実施の形態１を示す関数の再入可能性判定装置のソフトウェアブロック図である。

【図３】　この発明の実施の形態１の処理手順を示すフローチャートである。

【図４】　リソース排他制御の手段を表す概式図である。

【図５】　この発明の実施の形態２を示す関数の再入可能性判定装置のソフトウェアブロック図である。

【図６】　この発明の実施の形態２の処理手順を示すフローチャートである。

【図７】　この発明の実施の形態３を示す、関数の再入可能性判定装置のソフトウェアブロック図である。

【図８】　この発明の実施の形態３の処理手順を示すフローチャートである。

【図９】　この発明の実施の形態４を示す、関数の再入可能性判定装置のソフトウェアブロック図である。

【図１０】　この発明の実施の形態４の処理手順を示すフローチャートである。

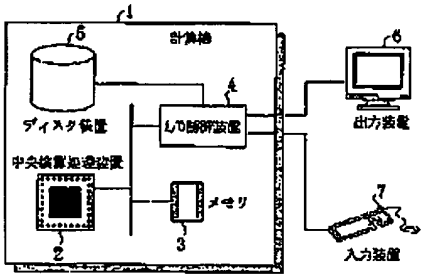
【符号の説明】

１　計算機、２　中央演算処理装置、３　メモリ、４　Ｉ／Ｏ制御装置、５ディスク装置、６　出力装置、７　入力装置、８　実施の形態１におけるリソース排他制御の手段の記述、９　実施の形態１におけるリソース定義、１０　実施の形態１におけるプログラミング言語の文法、１１　実施の形態１における再入可能性判定プログラムを出力するコンパイラ、１２　実施の形態１における対象関数を含むソースコード、１３　実施の形態１における対象関数が参照するリソースの定義を含むソースコード、１４　実施の形態１における再入可能性判定プログラム、１５　実施の形態１における判定結果、２６　関数定義、２９　排他制御領域、３２　実施の形態２におけるリソース排他制御の手段の記述、３３実施の形態２におけるリソース定義、３４　実施の形態２におけるプログラミング言語の文法、３５　実施の形態２に

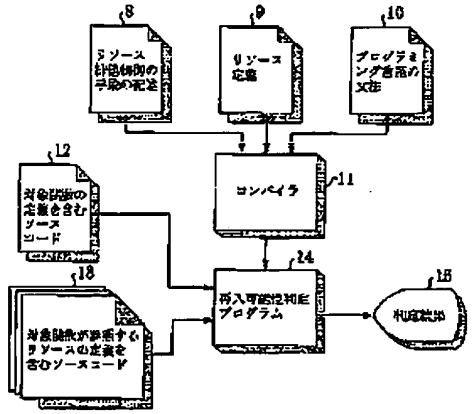
おける再入可能性判定プログラムを出力するコンパイラ．３６　実施の形態２における対象関数を含むソースコード、３７実施の形態２における対象関数が参照するリソースの定義を含むソースコード．３８　実施の形態２における再入可能性判定プログラム．３９　実施の形態２における判定結果、４０　実施の形態２における再入不可能原因出力装置、４１実施の形態２における再入不可能原因、５４　実施の形態３におけるリソース定義．５５　実施の形態３におけるプログラミング言語の文法．５６　実施の形態３における再入可能性判定プログラムを出力するコンパイラ、５７　実施の形態３における対象関数を含むソースコード、５８　実施の形態３における対象関数が参照するリソースの定義を含むソース＊

＊スコード、５９　実施の形態３における再入可能性判定プログラム、６０　実施の形態３における判定結果、６５　実施の形態４におけるリソース定義、６６　実施の形態４におけるプログラミング言語の文法、６７　実施の形態４における再入可能性判定プログラムを出力するコンパイラ、６８　実施の形態４における対象関数を含むソースコード．６９　実施の形態４における対象関数が参照するリソースの定義を含むソースコード．７０　実施の形態４における再入可能性判定プログラム．７１　実施の形態４における判定結果、７２　実施の形態４における再入不可能原因出力装置、７３　実施の形態４における再入不可能原因。

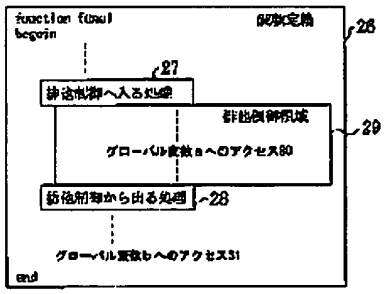
【図１】



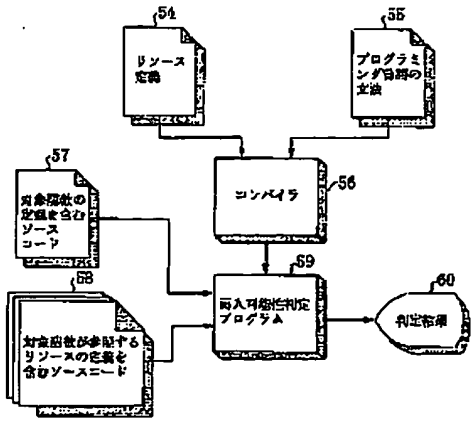
【図２】



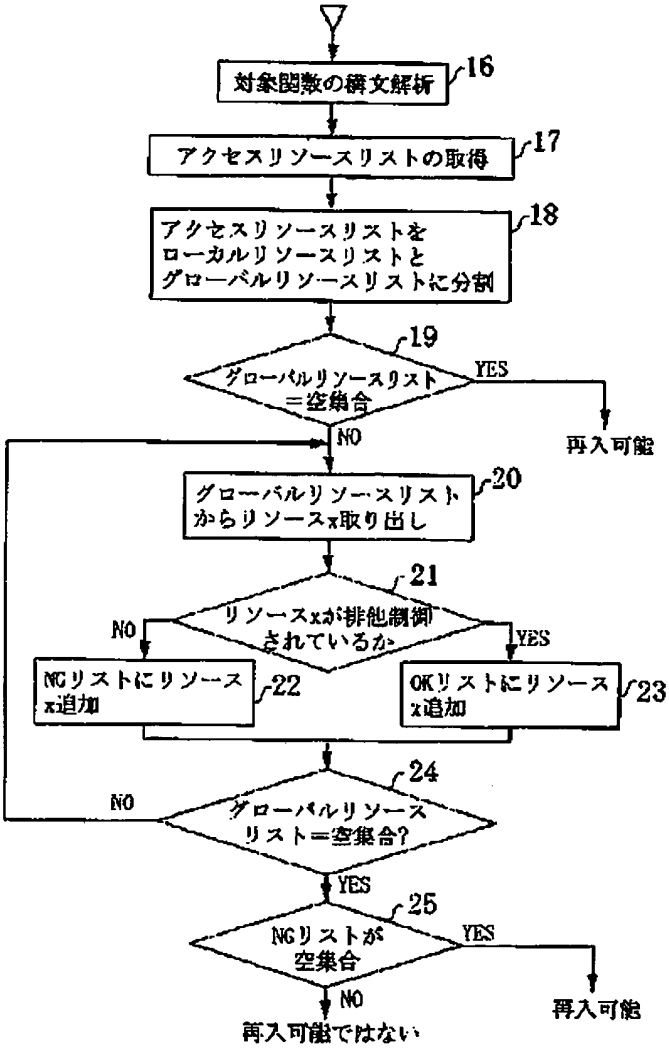
【図４】



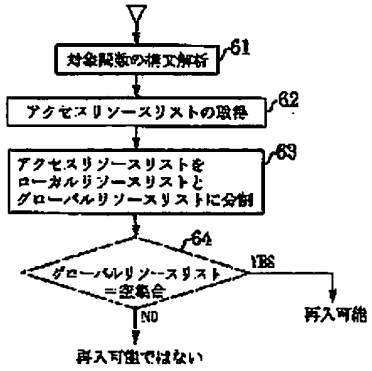
【図７】



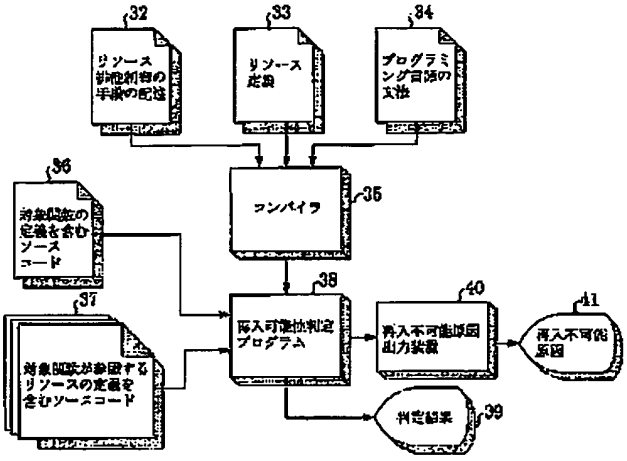
【図３】


対象関数の構文解析
16
アクセスリソースリストの取得
17
アクセスリソースリストを
ローカルリソースリストと
グローバルリソースリストに分割
18
グローバルリソースリスト
＝空集合
19
YES
再入可能
NO
グローバルリソースリスト
からリソースx取り出し
20
リソースxが排他制御
されているか
21
NO
YES
NGリストにリソース
x追加
22
OKリストにリソース
x追加
23
グローバルリソース
リスト＝空集合?
24
NO
YES
NGリストが
空集合
25
YES
NO
再入可能
再入可能ではない

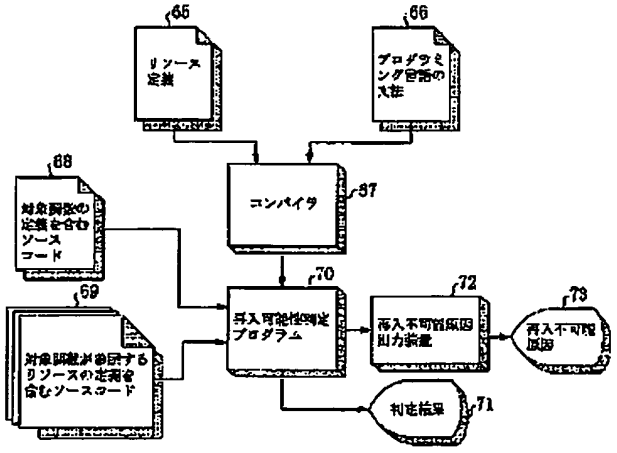
【図８】


対象関数の構文解析
61
アクセスリソースリストの取得
62
アクセスリソースリストを
ローカルリソースリストと
グローバルリソースリストに分割
63
グローバルリソースリスト
＝空集合
64
YES
再入可能
NO
再入可能ではない

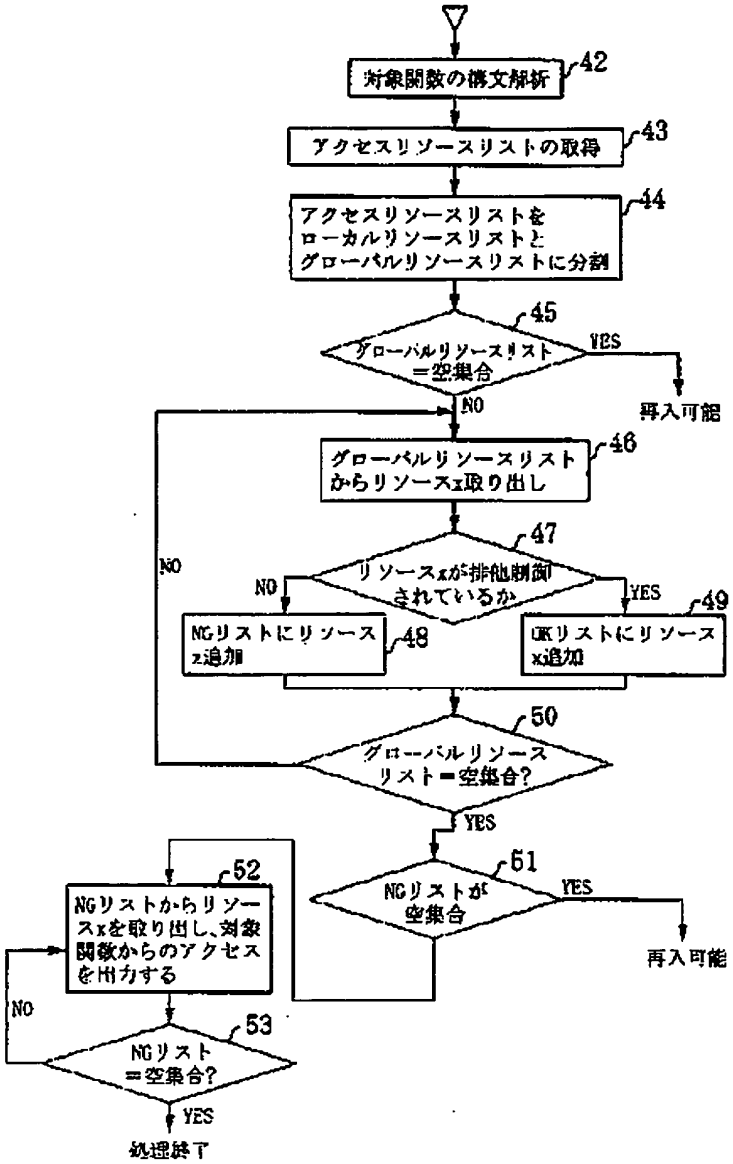
【図５】


32
33
34
リソース
定義
プログラミング言語の文法
36
対象関数の定義を含むソースコード
コンパイラ
35
37
対象関数がお使するリソースの定義を含むソースコード
再入可能性判定プログラム
38
40
再入不可能原因出力装置
41
再入不可能原因
判定結果
39

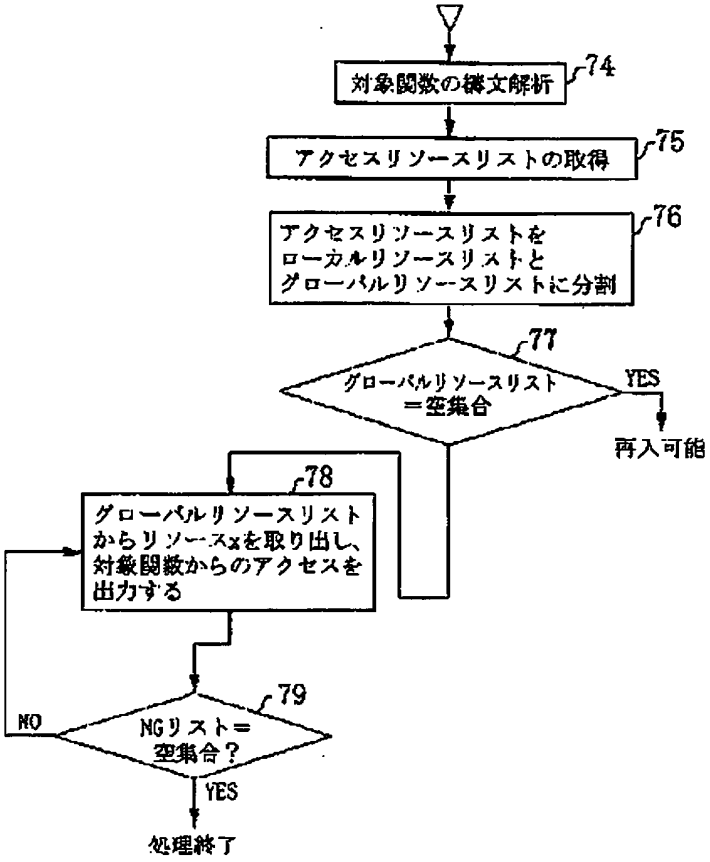
【図９】


65
66
リソース
定義
プログラミング言語の文法
68
対象関数の定義を含むソースコード
コンパイラ
67
70
69
対象関数がお使するリソースの定義を含むソースコード
再入可能性判定プログラム
72
再入不可能原因出力装置
73
再入不可能原因
判定結果
71

【図6】


対象関数の構文解析
42
アクセスリソースリストの取得
43
アクセスリソースリストを
ローカルリソースリストと
グローバルリソースリストに分割
44
グローバルリソースリスト
＝空集合
45
YES
再入可能
NO
グローバルリソースリスト
からリソースx取り出し
46
リソースxが排他制御
されているか
47
NO
YES
NGリストにリソース
x追加
48
OKリストにリソース
x追加
49
NO
グローバルリソース
リスト＝空集合?
50
YES
NGリストが
空集合
51
YES
再入可能
NGリストからリソース
xを取り出し、対象
関数からのアクセス
を出力する
52
NO
NGリスト
＝空集合?
53
YES
処理終了

【図１０】


対象関数の構文解析
74
アクセスリソースリストの取得
75
アクセスリソースリストを
ローカルリソースリストと
グローバルリソースリストに分割
76
77
グローバルリソースリスト
＝空集合
YES
再入可能
78
グローバルリソースリスト
からリソースxを取り出し、
対象関数からのアクセスを
出力する
79
NO
NGリスト＝
空集合？
YES
処理終了